Institute for Scientific Computing Research

Student Internship Research Summaries

siMPI: Serial Implementation of MPI

Maria Luisa M. Alano
University of San Francisco

Summary

The need has arisen for creating a sequential implementation of the Message Passing Interface (MPI) that conforms to the high-performance parallel version. Named siMPI, this project enables laboratory users to run MPI programs outside the POE environment of the IBM ASCI machines for debugging and testing purposes. This is convenient for developers and avoids allocation and waste of high-end resources. In addition, from a general perspective, siMPI is a tool that lets developers create parallel programs without the need for actual parallel machines. A single processor suffices, without the overhead resource allocation, trouble, and complexities.

To implement siMPI, the standard C programming language was chosen. The essential calls of MPI (the most common two-thirds of the entire 209 function-interface), such as MPI_Send, MPI_Receive, have been implemented so far. The Fortran equivalents have also been generated.

The behavior and output of the library calls comprising siMPI have been patterned, but not exactly mirrored, after the standard MPICH implementation.

The future for siMPI is promising. Once it is deemed stable and fit, siMPI will be packaged with the standard pyMPI so users may freely explore and use MPI with convenience.

The remaining one-third of the standard MPI calls should not be difficult to complete. Also, as with any software project, siMPI should undergo rigorous testing before release.

Multigrid Performance on Systems of PDEs Using Computational Local Fourier Node Analysis

David Alber

University of Illinois at Urbana-Champaign

Summary

Geometric multigrid is a fast and cheap alternative to Krylov iterative methods for some structured problems, but it is not always clear how well it will perform on a given problem. Surprisingly slow convergence rates sometimes plague a given problem. Multigrid performance on linear systems arising from systems of partial differential equations (PDEs) is understood theoretically for many special problems. This research aims to generalize the systems of PDEs whose multigrid convergence can be understood theoretically.

Local Fourier Mode Analysis (LMA) is a powerful technique that is applied to estimate smoothing and convergence rates of multigrid smoothers and methods. In addition, it is possible to use this technique to determine the spectrum of a preconditioned matrix. This is useful to estimate the effectiveness of using multigrid as a preconditioner on a particular problem. Therefore, LMA was selected as the method to use for gathering data. Unfortunately, very few software packages exist that perform LMA, and doing the analyses by hand is not an option. Therefore, a new software package was written to carry out LMA calculations. Using a library provided by Hiptmair and Metzger we authored an application that calculates smoothing and convergence rates. It also calculates the spectrum of a matrix preconditioned by multigrid. Other tools were added that make using the software more intuitive, such as a function that builds the stencil input files, and a graphical user interface.

This software is now being used within CASC to study different systems of PDEs amid how multigrid performs when trying to solve or precondition them. I hope to apply some of this work to a Masters thesis. I intend to expand the current software to offer more functionality, and, in any case, I plan to continue my collaboration with CASC on the research that has been started. Currently, we are working on a journal article that includes some of this work.

Wrapper Integration within the DataFoundry Bioinformatics Application

John C. Anderson
University of the Pacific

Summary

The DataFoundry bioinformatics application was designed to allow scientists to interact directly with large datasets, gathered from multiple remote data sources through a interactive graphical interface. This type of application is becoming essential, as there are more than five hundred sources of publicly available bioinformatics information available today. The need to access even a small fraction of the publicly available, heterogeneous bioinformatics sources on the World Wide Web motivates the development of a data access solution that facilitates use of external data.

Gathering information from multiple data sources, integrating that data, and providing an interface to the accumulated data is non-trivial. Advanced computer science techniques are required to develop a solution. One possible solution, which was applied to this problem, involves the use of wrappers. Wrappers are specialized information access programs that are able to transmute that information to a form usable by a single application. Wrappers were deemed the most appropriate way to extend the DataFoundry bioinformatics application to support data integration from multiple sources. By adding wrapper support into the DataFoundry application, we expect that this system will provide a single access point to bioinformatics data for scientists.

To date, we have been successful in defining and implementing a communication path between the client and server DataFoundry bioinformatics application programs that includes support for wrapper-based queries. Adding more external bioinformatics sources to the DataFoundry application is now as simple as developing a wrapper for a particular data source.

There are still limitations within the current framework. The most important is that there are currently relatively few wrappers available for bioinformatics data sources. Work continues to develop an automated wrapper generation system able to generate wrappers for any bioinformatics source. The second is performance; with relatively few wrappers, the client and server are able to process the volume of applicable data with ease. Once all of the relevant sources are wrapped, however, performance issues will become pressing.

Implementing the Column Inventory Algorithm

Cheryl Barkauskas
University of Wisconsin-Madison

Summary

One way to evaluate the accuracy of a climate model is to test it against known benchmarks, such as predictions of the amount and flux of dissolved inorganic carbon in the oceans. Calculating the column inventory, the amount of a substance present in the ocean under a certain surface area, is such a test.

Climate model data sets are sufficiently large (often over a hundred thousand points) that speed can be a factor in calculations. This makes the choice of programming language important, to minimize execution time. However, the functionals we are creating are tools for climate scientists, not computer scientists, so the simplicity of the interface to the methods is also a consideration. Therefore, several programming languages - Python, Fortran, and C - are tested to determine the best one in terms of execution speed and ease of implementation.

We first wrote a Python program that read a data file and extracted the necessary variables. This separates the interface from the calculation and provides each of the three implementations with a common standard to follow. We next created a separate set of Python calls to calculate the column inventory and mean. Finally, we translated the code from Python into C, and then from C into Fortran. (To communicate with the Python code, the Fortran module had to be compiled by a tool named Pyfort, a Python-Fortran connector developed by Paul Dubois at LLNL.)

Three working implementations of the column inventory algorithm - in Python, Fortran, and C - were completed. The Python version is most likely to be used by the climate research group at LLNL because most of its members understand Python, so it will be easy to deal with the source code. However, preparing the Fortran was also useful because it taught us the basics in making Python and Fortran communicate. Many of the group's programming routines are already written in Fortran, so the ability to call them from a Python interface is valuable. Speed of execution is not a major factor with current dataset sizes, even though Python generally runs about four times slower than either C or Fortran.

Additional features added to the basic algorithm include correcting volumes of partial bottom cells and applying regional (basin) masks to the data.

The climate group may now compare the output of the column analysis code to ship-track data to determine how well their models match the real world. Also, the versions will be archived for future reference for how to code Python-C and Python-Fortran programs.

We brought our summer research through the lab “review and release” process with a poster presentation at the 2002 student research symposium.

Improving the Network Intrusion Detector

Bridget Benson
Cal Poly San Luis Obispo

Jessica Fisher
Harvey Mudd College

Ian Webb
Colorado State University

Summary

Computer Security has become an increasing concern in today's networked society. The College Cyber Defenders (CCD) program creates a pool of young computer scientists knowledgeable about computer security. Our project was to enhance the Network Intrusion Detector (NID) by adding signatures to its database and updating the web-based Intrusion Detection Exchange Server (IDES). We also worked on finding a reasonable network setup for future CCD programs to use.

First, we created an internal network on which to run attacks, one that was not connected to the lab network. We also modified the Flexnet server developed at Sandia Lab to be a backup server through the Network File System. This system allowed us to recover damaged systems to a known state.

Next, we downloaded exploit code from the internet, wrote it to CDs to bring into our internal network, and attempted to run the exploits against our victim machines. We listened to the traffic with Ethereal, a graphical utility similar to tcpdump. From this traffic, we found packets which contained unique strings to identify the attempted exploits; these strings became new NID signatures. Most of the exploits we downloaded actually did not work, and we were able to obtain only 23 new signatures by this method. We then turned to an open-source intrusion detector, Snort, and wrote a PERL program to convert Snort's "rules" to NID signatures. Through this program, we obtained over 800 more signatures.

Our final task was to modify IDES to run with the Apache webserver. IDES provides a way for NID users to interact with developers and suggest improvements to the system, and offers utilities to create configuration files for NID and add signatures to the database. IDES was designed to run with Microsoft's IIS webserver, using Microsoft Access as its database and ASP (Active Server Pages) as its script language. To make IDES less platform-dependent, we modified it to run with the Apache webserver, using MySQL as its database and PHP as its script language. We also modified the PERL program to place the new signatures directly into IDES's database.

Future CCD programs' network setup would be enhanced by the use of a one-way file transfer system, which would help avoid the necessity of using CDs to transfer data. Enhancements to Flexnet to turn it into an easy-to-use backup server would also be beneficial. IDES would be improved by adding functionality to assist more of NID's multiple features.

Improvements to ALE3D

Timothy Campbell
University of Arizona

Summary

We implemented five improvements in the Livermore workhorse code ALE3D.

ALE3D employs a point-sampling algorithm to determine volume fractions in elements composed of mixed materials. We devised and implemented an algorithm that efficiently calculates these volume fractions explicitly, and in comparable run-time. This will result in better simulation accuracy, especially for input with multiple materials that mix.

ALE3D was not capable of material burn across slide surfaces (partially disconnected regions of the mesh). I altered ALE3D's burn routines to detect slides, and burn across, assuming a burn material exists on the other side, and to do so in such a way so as to correct for disparate mesh zoning across the slide.

We introduced Python hooks into ALE3D. The code is now capable of reading in Python source code as input to modify internal data structures at run-time. This allows users, for example, to create boundary conditions using arbitrary functions at arbitrary time steps.

We expanded ALE3D's internal mesh generation capabilities. ALE3D previously relied on externally generated meshes, but can now create simple meshes based on geometries like spheres, cones, blocks, etc. This makes generation of test problems considerably less time-consuming, both in terms of problem generation time and overhead associated with learning how to generate a mesh.

We modified ALE3D's input code to allow it to read in large data files in parallel. Mesh input is read and sent only to nodes that need it, based on an a priori domain decomposition of the problem.

These changes to ALE3D range in type from improved accuracy, greater functionality, greater ease of use, and improved performance.

Helios: Illuminating C++ Memory Management

Karl Chen
UC Berkeley

Summary

Helios is a memory-allocation software tool that provides a flexible and efficient way of debugging large C++ programs and Python extensions in a developer-friendly, user-steerable way. It is intended to be used in debug mode, as well as production code. Helios allows dynamic (run-time) control of optimization and debug options. If necessary, these options may be fixed at compile time for maximum optimization (each option fixed is one less memory access and/or conditional branch at runtime).

Helios contains a “Controller” (C++) that delegates allocation and deallocation to the appropriate “Allocator”. The various “new” and “delete” operators (C++) and malloc() and free() (C) call the Controller's allocate() and deallocate() functions. The Controller can be manipulated directly in C++ or through Python (with Pyffle wrappers). The first time we ran Kull with Helios, Helios immediately caught a certain kind of memory bug (malloc/delete mismatch).

Kull, the project Helios was created to support, and other lab projects are planning on using Helios, and we will personally continue to maintain it. The code for Helios will be put up for software Review and Release.

Draco Help System

John Clark
Northern Arizona University

Summary

Draco is a PMesh physics model program used to create meshes. Draco integrates an OpenGL graphical interface to display the mesh, as well as a command line interpreter used to manipulate the Python objects that control the mesh. The Draco help system provides the user with a localized place for finding documentation on all of the Python objects available. The documentation will inform the user of all the methods, functions, classes, and data that a particular object has, and it is organized in an easy to read website format.

PyDoc is a documentation tool for the Python scripting language, it will get the information about any Python objects and output it into a text format. Unfortunately, PyDoc does not have the ability to create the beautiful html format that we wanted. However, the C++ and C documentation tool Doxygen does have the ability to create html as well as many other desirable formats of output. So we created a bridge between the two. The bridge uses PyDoc to get the information from the Python objects in Draco, then it formats that information to create files that Doxygen can understand. After this, Doxygen can be used to create the html pages.

As another part of the project, we made viewing help pages easy to do while using Draco. Since the pages are in html, a browser is the best way to display them. We allowed for two options. The first option is an integrated browser written with the QT windowing library; the second option is to use a browser integrated into the operating system environment. The integrated browser is very simple and does not allow for things like style sheets, or backgrounds with differentiated highlight colors. The system browser can be anything like Netscape, Mozilla, or Konqueror, and is determined by setting the environment variable DRACOBROWSER to the browser of choice. Either of these browsers may be launched, while running Draco, from the command line interpreter or from the Help pull-down menu in the Draco GUI.

The help system will be maintained and augmented by my supervisor who did the initial work on the bridge between PyDoc and Doxygen.

Computations in Scalar Field Topology

Kree Cole-McLaughlin
University of Utah

Summary

Scalar fields are used to represent data in different application areas like geographic information systems, medical imaging or scientific visualization. One fundamental visualization technique for scalar fields is the display of isocontours - that is, sets of points of equal scalar value, also called the isovalue. For example in terrain models isolines are used to highlight regions of equal elevation. The Contour Tree is a graph that represents the relations between the connected components of the isocontours in a scalar field. Two connected components that merge together (as one continuously varies the isovalue) are represented as two arcs that join in a node of the graph. Each node occurs at a critical point in the scalar field. For a simplicial domain with a piece-wise linear interpolant, Carr et al. present a simple algorithm for computing the Contour Tree with complexity 0(n log n + N), where n is the number of points in the domain, and N is the number of simplices.

One fundamental limitation of the Contour Tree is the lack of additional information regarding the topology of the contours. The topology of an isocontour is fully characterized by its Betti numbers. In 3D fields the isocontours are surfaces and the Betti numbers correspond to the number of connected compo-nents, the number of tunnels, and the number of voids enclosed by a surface. Pascucci has presented a simple and fast algorithm for the construction of the Contour Tree of a 3D scalar field augmented with the Betti numbers of each contour. The complexity of this approach is 0(N log N) in time and 0(N) in storage. The nodes of the Augmented Contour Tree correspond precisely to every critical point of the field. Thus we trade a slightly slower algorithm for more information.

We have implemented both of the above algorithms and produced images of complex contour trees for a few data-sets. The images are produced with the graphiz utility from AT&T. Our data-sets are quickly approaching the limit of this utility's ability. Therefore in the future we will need to develop our own tool for drawing the Contour Tree. We have designed a third algorithm that computes the Augmented Contour Tree in 0(n log n ± N) time. This new algorithm combines the efficiency and simplicity of the scheme of Carr with the augmented information of Pascucci. Thus we have successfully introduced new and useful information to the Contour Tree data structure without slowing down the computation.

Our main result is a completely new approach for computing the Contour Tree for rectilinear grids. This scheme is based on a divide-and-conquer approach, and assumes only that we can construct the tree for a single cube. In this method we are not limited to using piecewise linear interpolation, which can introduce false topology on rectilinear grids. In the time I have spent here I have implemented all of these algorithms, and have collected several practical results. The time complexity of our scheme is further

Kree Cole-McLaughlin
University of Utah

Summary (continued)

improved to 0(n + t log n) where t is the number of critical points of the mesh. We have applied this technique to the methane data set of F. Gigi.

We have several goals for future development of this project. The method of producing the Contour Tree of a cube with a trilinear interpolant involved a careful study of the topology of the trilinear function. For example, we discovered a case missed by earlier papers classifying the possible configurations of the trilinear interpolant. We plan to publish our new result with a formal proof of the completeness of our analysis.

In the possible extension of this project we plan to study the practical scal-ability of the algorithm for very large data-sets. We also plan to implement a parallel version of our new algorithm and develop a new GUI component for the effective display of the Contour Tree.

Late Time Outgoing Distribution from Spheres

Hillary E. Davis
Georgia Institute of Technology

Summary

The dominant radiation transport model used for modeling Inertial Confinement Fusion (ICF) is Implicit Monte Carlo (IMC). Although there exists an acceleration scheme that provides a factor of five increase in performance, its use is restricted to well defined problems. In this project, we are reducing the bias present in the acceleration scheme so that there are no limitations on its use.

The overall project comprises two methodologies: Compressed Walk (a non-recursive solution to the random walk of a constant speed particle) and adjusting diffusion solutions. Our effort was to find the thickness dependent late time outgoing distribution to be used in concert with the diffusion methodology.

We chose a thickness for the media and placed an isotropic point source in the center. Subsequently, using Monte Carlo, we followed photons as they incurred isotropic scatters. As the photons left the surface of the media, tallies were made. These numerical experiments recovered not only the distribution and its error, but also several velocity moments. The distribution was then fitted to a polynomial of the given form and tested against measured moments.

We were able to capture the distribution and moments in less than thirty coefficients. Such a short polynomial makes our scheme practical for use in reducing bias, without incurring a significant runtime penalty.

The angular distribution can be broken down into four regions: early time internal and outgoing, and late time internal and outgoing. We have presented the late time outgoing. A preliminary investigation into the data suggests that the early time outgoing and the late time internal are tractable. Although we can measure the early time internal behavior, the complexity of the distribution makes it difficult to gain runtime advantage over the Compressed Walk solution. Hence, work will continue on the time dependent outgoing and late time internal distributions as well as the Compressed Walk solution.

Tools for Parallel Finite Element Analysis

Paul Dostert
UC Davis

Summary

Computing numerical solutions to finite element problems in parallel can often be difficult task. Our goal was to create tools with which users can simplify the process, from creating a mesh all the way to solving the finite element matrices in parallel.

The natural first step in finite element problems is to define a mesh. For our code, Netgen was used to create triangular or tetrahedral meshes. After a mesh is established, we use the Metis mesh partitioner to split the mesh into multiple subdomains, one per processor. The mesh information, as well as the connectivity between subdomains, is written to a different file for each subdomain.

Each processor reads in the mesh information corresponding to its subdomain, and the finite element matrix is created according to its degrees of freedom on the local subdomain. Next, a matrix corresponding to the local-to-global degree of freedom table is created on each processor. This matrix, which we will call P, as well as the finite element matrix, A, are stored in ParCSR format. The ParCSR format was chosen due to the efficiency which it stores sparse matrices in parallel, and its compatibility with the well established hypre routines. To obtain the full finite element matrix, across subdomains, we take P^T A P. We then apply one of many possible linear solver routines from the Hypre toolkit to generate the solution on each processor.

Results were satisfactory, with some routines scaling at near perfect levels. If input time is included, the total time to input and solve the system was actually greater than two times faster on double the number or processors in some cases. This is an I/O artifact: the input files are smaller if we split a domain into more subdomains, lowering the input time on each processor.

Future development of this code may involve improving compatibility with different mesh generators, including generators that use non-triangular or non-tetrahedral elements. Other generalizations include implementing other parallel matrix storage formats and investigating their benefits.

Efficient Translators for Document Object Model Trees

Roger Elion
Purdue University

Summary

Cyclops is a program written in Java that allows users to utilize a graphical user interface to create or make changes to legacy physics codes. A second program, written in Python, converts a language-specific parameter description file into a data structure that will automatically and dynamically build another data structure that represents an input deck. Upon parsing the input deck, each value and comment is added to a Document Object Model (DOM) tree, which is then output in XML and passed to Cyclops for the user to interface.

This input translator is a far more condensed and efficient than the existing language parser, written in Perl, that was being used. The concept of recycling the parameter description file in order to create a look-up hash automated this portion of the process. (This was the idea of Thomas Richmond, a returning summer student.) In addition to this, the recently released DOM tree structure enables programmers to convert DOM trees to XML by merely calling a function. Overall, the decision to incorporate these fairly young languages with easy to manipulate data structures has condensed thousands of lines of Perl code into hundreds of lines of Python.

Increasing the Transfer Speed of SSH to Utilize Full Network Bandwidth

Jason R. Estrada
Baylor University

Summary

This report presents a performance analysis and describes several techniques to obtain more performance from the secure networking tool SSH, including implementing a high performance protocol, using multiple processors to perform encryption, and making use of multiple sessions. A solution, using multiple SSH sessions to multiplex data transfer over the network has been developed in an effort to gain faster transmission speed over the network, while still using strong encryption ciphers.

SSH is a protocol for secure remote login and other secure network services over an insecure network. SSH has become the de facto standard for obtaining a remote shell or transmitting data across the network securely in a day in which security is an increasing concern. However, we would also like to accomplish this very quickly.

Most of the processing time spent within SSH is in the actual encryption of the packets, and not in the transfer of the packets across the network. It, of course, takes time to send the packets over the network, but that time pales in comparison to the time spent in the encryption algorithms. Since SSH (at least in the case of OpenSSH) is not multithreaded, CPU cycles are often dormant on multiprocessor machines that could be used to perform encryption/decryption on the outgoing/incoming packets.

All testing and benchmarks were performed in a non-routable subnet containing two host machines with a round trip time (RTT) of 116 usec, each running Redhat Linux 7.3. Neither host was mounting NFS or AFS, in order to minimize the network traffic over the line. A set of benchmarks was performed on the systems and the network to discover performance and scaling metrics. Netperf was used to measure the maximum achievable throughput of the network

The Secure Shell used in the testing was from OpenSSH. At the time of testing, the most recent version of OpenSSH was 3.2.3pl. OpenSSH can be obtained at www.openssh.org. Only the SSH2 protocol was used and the SSL library used for encryption was provided by OpenSSL's distribution of SSL. At the time of testing, the most recent version of OpenSSL was 0.9.6d. OpenSSL can be obtained at www.openssl.org. Public/Private key authentication was used since SSH was compiled and run within user space, instead of running as the system. Although not recommended, a null key phrase was used on the keys so it would not be required to enter a password. This made the testing of SSH more feasible.

Netperf is a benchmark utility that can measure end-to-end network latency and throughput. The latest version at the time of benchmarking was 2.lpl3. Netperf can be found at www.netperf.org. The results from these two test show that the maximum available throughput of the 100Mb Ethernet line is approximately 94Mb/sec with 99%

Jason R. Estrada
Baylor University

Summary (continued)

confidence. No TCP optimizations were applied to the network, such as TCP tuning, since no root access to the machines was available.

Testing the transfer speed of SSH was achieved by using two shell scripts, ssh2_test. sh and full_ssh2_test. sh. The shell script full ssh2_test. sh was used to test all the supported SSH2 ciphers. The shell script logged the results to individual log files for later examination. Minor additions to the SSH source code were made to time and calculate the effective transfer rate of data across the network without using an external program. The shell script assumes that it is executing the modified binary of SSH; otherwise no output will be displayed in the log files.

Efficient Approximation of Solutions to Mass Matrix Equations

Aaron Fisher
UC Davis

Summary

The finite element method approximates solutions to partial differential equations on arbitrary domains tesselated with a mesh. The bulk of the CPU time involved in computing this approximation comes from repeatedly solving a linear system, Ax=b, composed of a mass matrix A, a vector of the degrees of freedom x, and a right-hand side b. The mass matrix, which is dependent on the geometry of the mesh and the order of the approximation, tends to be sparse, but can be quite complicated for unstructured meshes. Furthermore, in problems where the mesh is allowed to deform, the mass matrix changes at each time step to accommodate the deformation.

Since mass matrix solutions dominate the execution time, their CPU requirements make a natural target for amelioration in finite element problems. Even relatively small improvements in these solves prove valuable since they appear in each time step.

Our project focused on approximations to the mass matrix equations for an optical fiber waveguide problem. By comparing the results to an analytical solution to the optical waveguide problem, error values were obtained for the various approximations. We studied both this discretization convergence behavior and algebraic convergence behavior for the iterative solution of the resulting system.

Specifically we studied the preconditioned conjugate gradient with block Jacobi, banded, and ILU preconditioners, with variations in block size, band radius, and fill levels. We also studied a well-known approximation to the mass matrix known as mass lumping.

Through this work we gained a valuable understanding of the error levels involved in this class of approximations. We also found that the convergence behavior of the methods is sufficiently well behaved to consider loosening up the convergence tolerances in order to save CPU time. Finally, we found that by adjusting the block size/fill levels in the block Jacobi/ILU preconditioners we could save a moderate amount of CPU time.

Despite some modest successes, this remains a work in progress. Future plans include propagation of linear system error into the overall simulation in order to determine how much is tolerable, and experimenting with more esoteric solution methods.

Compression of Scalar Functions

Ilja Friedel
Caltech

Summary

Scientific simulations generate huge data sets, which need to be transmitted and stored for future analysis. To reduce size and transmission time of the generated files, compression schemes are employed. Current methods are often based on wavelets defined on grids. These methods perform well on smooth data but loose efficiency near discontinuities. It is known that meshes that adapt size and shape to the data have better approximation properties than regular grids. Unfortunately the storage overhead for arbitrary meshes is high compared to regular grids. It is our goal to create a progressive, triangle based compression method for 2d slices of data. Progressivity permits browsing of data sets at coarse resolution without need of transmitting the whole file.

It was our goal to avoid the overhead associated with the transmission of connectivity information. For this reason in our model the receiver makes data dependent decisions (based on the already received data points) on the construction of the mesh. The sender monitors these decisions by running the same construction.

At each iteration of the algorithm
- the receiver selects the triangle with the expected highest error
- a point inside this triangle is chosen, such that the insertion is expected to provide the largest error reduction
- the difference to the prediction of the z-coordinate of this point is transmitted from the receiver
- the selected triangle is split using the new point
- based on this new knowledge the receiver re-triangulates the mesh to obtain a good approximation for the next step

We examined different strategies for each of the steps in this algorithm and compared them using rate distortion curves. Most of these strategies require the construction of a local quadratic model of the surface based on discrete curvatures. Point insertion based on a modified farthest point strategy combined with edge flipping based on bending energy where shown to be efficient for many smooth input data sets. With the new algorithm approximation errors of 35% to 50% of the magnitude of grid based errors where achieved when using the same number of data points.

To get a practical method the residuals of the transmitted data points need to be compressed. We are considering different bit-encoding schemes.

The current algorithm can be classified as a lossy method, because it assumes continuous data. In many applications the data is defined on a grid. By choosing the inserted points from this grid the algorithm will operate in a lossless mode.

In our implementation only residuals are transmitted. We plan to examine heuristics for the sender to guide the decisions made by the receiver.

Exploiting Workload Distributions for Application-Level Dynamic Load Balancing

Karen Glocer
UC Santa Cruz

Summary

A large volume of research in distributed computing suggests that load balancing is a primary concern, and it will inevitably become more difficult as applications get larger and more complex. The goal of our work is to allow automated load balancing at the application level. Target applications are typically implemented in MPI. MPI has been described as the assembly language of parallel programming. Like assembly language, it is both efficient and difficult to manage. For years people wrote in assembly for its efficiency, but eventually it was superseded by high-level languages through the advent of optimizing compilers. MPI is currently the standard for high performance scientific applications, but as problems increase in complexity and size, some difficulties will become more evident and it will benefit developers to look at handling these difficulties with higher-level parallel programming models.

Most previous load balancing research falls into three categories: scheduling between independent jobs, workload characterization, and balancing the load within an application, something that is usually handled by the application programmer. Only relatively recently has automated load balancing been proposed for balancing load within the application, and not just between independent applications. The two most notable examples of this are Charm++ and Cilk, both of which use a programming model in which work is divided into discrete chunks where the number of chunks is much greater than the number of processors. These chunks can then be moved around based on their estimated computational and communication costs.

In these systems, computation and communication costs are estimated heuristically and no attempt is made to characterize the workload distribution of real scientific applications. Our work fills this gap. Using a new tool, the Sequoia Tracer, we gathered traces of four applications: sPPM, SWEEP3D, High Performance LINPACK, and FLASH. These are each representative of programs run on the ASCI clusters and are common in scientific computing. In particular, FLASH uses adaptive mesh refinement, a technique that is growing in popularity and which requires a great deal of internal load balancing.

The traces contain each entry and exit from a computational state as well as communication. A trace is generated for each processor used by the application in question. Because tracing can itself affect timing information, it makes sense to use alternative metrics to measure the amount of work each processor is doing. In this case, a count of floating point operations is used. From this information, we were able to get the distribution of work done by each processor between communications. Each instance is analogous to a chunk of work in a programming model such as Charm++, so the distribution can serve as a way to estimate computational costs that were only estimated with heuristics in the past.

Karen Glocer
UC Santa Cruz

Summary (continued)

The results are surprising. The workload distributions for these chunks appear to be strongly bimodal. Most of the chunks perform very little computation, a few are very long, but chunks in between are infrequent. The small computational loads seem to fit a Pareto distribution in three of the four applications we tested. The distribution of chunks with large computational load is harder to classify because the numbers are smaller. Most importantly, these results show that it is very easy to predict which chunks will perform the most work, and these can be preemptively moved to either a dedicated processor or to one with a much smaller load, in a method similar to the one proposed by Harchol-Balter and Downey.

Interactive Exploration of Large Isosurfaces in Volume Datasets

Benjamin Gregorski
UC Davis

Summary

Numerical simulations performed on LLNL supercomputers are generating unprecedentedly large amounts of data. A standard method for visualizing, exploring, and understanding this data is to examine various isocontours. The problem with directly viewing the isocontours is that each consists of hundreds of millions of elemental triangles. Surfaces of this size cannot be viewed at interactive frame rates on conventional desktop machines. Intelligent preprocessing techniques and runtime algorithms are needed for interactive exploration and visualization. The goal of this project is to develop a system for dataset exploration on desktop workstations.

Large datasets are divided into subsets called bricks and stored on disk or on a remote system. These bricks are loaded and unloaded by the application as they are needed so that system resources are effectively utilized.

Interactive frame rates can be achieved using algorithms that employ multiresolution data structures with view-dependent rendering. Multiresolution data structures make better use of available storage space and computational power by representing important areas with more detail and less important areas with less information. View-dependent rendering selects what portion of the data to render based on where the user is looking and on the user's perception of the data. Objects outside the field of view do not need to be rendered, and objects that are far away can be rendered at lower resolutions. View-dependent rendering takes advantage of the multiresolution data structure to efficiently select what to draw. Our multiresolution data structure is a recursive tetrahedral mesh based on longest edge bisection. The mesh structure supports fast, local refinement necessary for view-dependent rendering. In addition, it supports an efficient method for ensuring mesh continuity required for iso-surface extraction. View-dependent rendering calculates the distortion on the view screen; it adjusts the refinement of the multiresolution mesh, selecting finer levels where more detail is needed and coarser levels where less detail is needed. Our continuing work is focused on working with large datasets, accurate representation of the data, and computational optimizations.

A paper documenting our summer work has been published at the 2002 IEEE Visualization Conference. It is also available as UCRL-JC-146819. Our continuing work is focused on working with large time-varying datasets, accurate representation of the data, and computational optimizations including occlusion culling, asynchronous rendering and compression.

A Parallel, Adaptive Implementation of the Immersed Boundary Method using SAMRAI

Boyce Griffith
New York University

Summary

The immersed boundary (IB) method provides a mathematical and computational framework for addressing problems involving fluid-structure interaction and has proved to be especially useful in simulating biological fluid dynamics. Realistic simulations that use the IB method require both high spatial resolution and very small timesteps. Consequently, high performance computing is an important component of simulation research employing the IB method.

Currently, all high performance IB software is designed to run on shared-memory vector machines. There is a clear need to develop IB software which will run efficiently on distributed-memory computers. We are harnessing the expertise available at CASC to develop such software.

The IB method specifies the interaction of a fluid, described as an Eulerian variable, and an elastic material, described as a Lagrangian variable. Consequently, the fluid is typically discretized on a Cartesian grid, while the elastic material is described by a network of Lagrangian points. A smoothed approximation to the Dirac delta function is used to connect these two quantities. Through the discrete delta function, quantities such as velocity may be interpolated from the Cartesian grid to the Lagrangian points, and quantities such as force or density may be spread from the Lagrangian mesh to the Cartesian grid.

High spatial resolution is required near the boundaries in order to capture important boundary layer flow. Since the current discretization of the IB method is very well suited for use of structured AMR, we are developing IB software using the SAMRAI library.

After one summer's work, single level IB software using SAMRAI and hypre is nearly complete. This software should allow for IB computations which make effective use of distributed-memory computational facilities.

Upon completion of the single level IB software, we will begin incorporating AMR into the solvers. Currently, there is no parallel implementation of the IB method that includes AMR. We are also interested in extending the IB method to coupled electrical-mechanical biological models.

We are also interested in developing new computational approaches for fully implicit temporal discretiza-tions of the IB equations. The current semi-implicit discretization typically requires very small timesteps due to the stiffness of the IB equations. An efficient implicit timestepping scheme could potentially ease this timestep restriction and allow for more realistic simulations.

Undecimated Wavelet Transforms for Image De-noising

Aglika Gyaourova
University of Nevada, Reno

Summary

Almost every kind of data contains noise. Noise reduction is a required step for any sophisticated algorithm in computer vision and image processing. This problem has existed for a long time and there is no general-purpose solution for it. A tradeoff always exists between the removed noise and the blurring in the image. The use of wavelet transforms for signal de-noising has been started in last decade. The capability of wavelets to render detail about spatial-frequency information is the main reason for this investigation. This property promises a possibility for better discrimination between the noise and the real data. Successful exploitation of wavelet transforms might lessen the blurring effect or even overcome it completely.

There are two main types of wavelet transforms – continuous and discrete. Because of the discrete nature of computers, computer programs exploit the discrete wavelet transform. The discrete transform is very efficient from the computational point of view in operation count and in compression of storage through decimation – the discarding of fine-scale data that is not needed to represent regions of coarse scale variation only. Its only drawback is that it is not translation invariant. Translations of the original signal lead to different wavelet coefficients. In order to overcome this and to get more complete characteristics of the analyzed signal, the undecimated wavelet transform was proposed. It carries out the full transform without decimating the signal. Thus, it produces more precise information for the frequency localization. From the computational point of view the undecimated wavelet transform has larger storage space requirements and involves more computations.

There are two classical algrorithms for computing the undecimated wavelet transform exist – “algorithm a trous” and Beylkin's algorithm. These algorithms approach the problem from different directions. Another class of undecimated algorithms has been discovered in a search for completely different characteristics. We have constructed three new undecimated algorithms and tested their performance for image de-noising. The experimental results have shown that these algorithms have better performance in the terms of noise the removal/image blurring ratio.

Mesh Quality Analysis

Matthew Haddox
University of the Pacific

Summary

Mesh generation plays a vital role in physics simulations. The quality of the mesh strongly correlates to the quality of the simulation: A problematic mesh can lead to inaccurate answers in simulations. It is therefore worth investigating the mesh quality before simulation, to avoid wasting time and resources on a simulation only to find inaccurate results – or worse, not to notice such results.

Three different classifications of algorithms were used. Individual element metrics from the Verdict library were incorporated, for both two- and three-dimensional meshes. Such metrics include aspect ratio, skew, taper, volume uniformity, area uniformity, stretch, diagonal ratio, oddy, condition, jacobian, scaled jacobian, shear, shape, relative size, shape and size, and others. Topological checks were also used. Measurements of node degree, and Eulerian conformal test algorithms were coded and incorporated. Interference checks were also developed to ensure the mesh was not tangled and elements did not overlap. Such tests were incorporated into the Visit visualization project, and the Draco mesh generation project. In such programs, elements of poor quality can be assessed, and then visually identified.

An investigation of poor metric correlations is being considered, and may be performed in the future. The current mesh quality tests will be used in multiple simulation projects to prescreen meshes before simulation.

Limited-view X-ray Tomography with the Stochastic Engine

Keith Henderson
Purdue University

Summary

HADES is a high-fidelity radiographic simulation code that can handle both mesh files and combinations of simple volumes such as spheres, ellipsoids, and cylinders. The Stochastic Engine is a code that works with a forward simulation code and experimental data to perform inverse reconstructions. It uses powerful statistical techniques to "guide" a random walk through the parameter space specified by the user. Together, HADES and the Stochastic Engine grant more powerful tomographic capabilities than traditional techniques, especially when the number of experimental radiographs is limited by experimental conditions.

A full Python installation, including several extension packages, was installed on the local Linux machine. The Stochastic Engine (a set of Python modules) statistical optimizer was also installed. Python modules were developed to vary input parameters to the forward model. The radiographic simulation code HADES was ported to the system and installed as the forward model into the Engine.

Concurrently, several metrics (e.g., Euclidean distance) for image comparison were developed and tested. The metrics are available to be incorporated into the Engine as user-selectable Likelihood Functions.

The combined simulation-optimizer was successfully tested with a single sphere, an ellipse-sphere Boolean object, and a helical post. Each object had at least two model parameters that were examined during the optimization process, and which were accurately reconstructed.

The integrated package will be used to analyze several experiments in which only two or three radiographs were taken. It is hoped that some expertise can be gained using the engine to estimate variances in various radiographic parameters, so that the corresponding image analysis can be made more quantitative.

An Adaptive Grid Method for an Ocean General Circulation Model

Aaron Herrnstein
UCDavis

Summary

The continuously increasing rate of CO2 emissions into the atmosphere is a growing concern of climatologists. It is hypothesized that an excess of such gases will produce a greenhouse effect thus increasing global temperatures and ultimately distorting global climate. As an alternative to releasing greenhouse gases into the atmosphere, it has been proposed to deposit them in the ocean. Major concerns of such "carbon sequestration" are: (1) the amount of CO2 that escapes back into the atmosphere and (2) any biological impacts brought about by changes in oceanic pH levels.

Ocean General Circulation Models (OGDM) are used to model sequestered carbon over a time scale of centuries. The OGCM used by LLNL's Climate and Carbon Cycle Modeling group shows discrepancies in pH changes between fine and coarse grids. An even finer grid is needed to determine convergence, but refining the entire grid will be quite costly in terms of computer time (on the order of months). Adaptive Mesh Refinement (AMR) allows desired sections of the grid to be refined, thus reducing run time considerably.

It is the goal of this research to produce an OGCM that uses AMR. Such a tool will be very useful for areas such as carbon sequestration. For implementing AMR, the LLNL-developed software package SAMRAI (Structured Adaptive Mesh Refinement Application Infrastructure) is used. Refinement of the grid on which the tracer field is hosted is based on gradients and possibly other criteria as needed. In addition, regions of critical topography may be refined if necessary.

Current tools developed are time integrators common to OGCMs but uncommon to AMR. The numerics implemented by these integrators include leapfrog, predictor-corrector, and Runge-Kutta. Tests were formed in 2D on a convection/diffusion model and on a shallow water model. Data for both cases was defined at cell centers. Work for the immediate future involves modifying the integrators to advance data on a staggered grid. Such grids are more typical of OGCMs which define momentum at nodes and all other quantities (pressure, temperature, salinity, etc.) at cell centers.

Surface Patch Trimming for Isosurface Visualization

W. Taylor Holliday
UC Davis

Summary

Very large time-varying datasets, such as the ASCI Turbulence PPM Simulations, will require temporally aware compression algorithms to facilitate interactive isosurface visualization. This may be accomplished, in part, by re-parameterizing the isosurface mesh to a form more compatible with compression and interactive display: a multiresolution mesh structure. The final steps in this process are the fitting of a subdivision surface to the re-parameterized mesh, and the compression of the surface. Surface patch trimming was explored as a way to “cut” into an isosurface, selectively revealing hidden features.

The fitting algorithm is based on the quasi-interpolation method of M. Duchaineau, which is simple and local. The algorithm has been extended to adaptively sacrifice locality for accuracy (a user-specified error tolerance is met) at lower levels of resolution, where improving the detail coefficients is computationally inexpensive.

The patch trimming algorithm is a variant of Warnock's algorithm, extended for view-dependant multiresolution trim curves. A toy implementation of the trimming tool was implemented for two-dimensional domains. A compression algorithm remains to be implemented.

The fitting algorithm should be tested with CIPIC's (http://graphics.cs.ucdavis.edu) re-meshing pipeline, and on more realistic data. A comparison with the wavelet approach of N. Litke should be made and filtering of the surface data should be explored. Creating temporally coherent parameterizations is also a topic of future research. Lastly, an interface for interactive trimming of surfaces should be implemented.

PMESH - CommData Checks

Bryan Hunter
Allegheny College

Summary

The goal of this project is to provide certain checks in the communication data output in the Kull-Lite data structure. There are certain problems with this aspect of Kull-Lite when running serial, on one processor, with multiple domains. The goal was to take the output and provide several checks on the data to locate where the possible error is present. This would give some insight on to where the problem would be located and hopefully aid in debugging.

The first task at hand was to understand the parallel programming aspect of the code and how to communicate between processors using nonblocking communication with the Message Passing Interface (MPI) in C++. Once a reasonable understanding was obtained of how that process worked, a study of the Mesh design and its representation in the code was needed. To accomplish this objective we used MPI in C++ for the parallel check, and were also capable of running these checks in serial. The first step is to verify that all sent and received data on the different domains matched in content, which consists of checking all the zones, faces, and nodes. The next priority is to check each domain to reassure that it only had data that it should own – no more and no less. Next, we confirm that all the data to be sent is on the domain boundary, adjacent to the data to be received. The final check is similar to the previous, except from the receive data side, checking to make sure all its data is on the domain boundary, adjacent to send data.

From some trial tests, this procedure has shown progress in finding regions of code where communication bugs occur. These checks will continue to be used and will provide a guide status report for when the data is created, displaying any possible errors located within it. The PMESH team has already found and fixed an error that was revealed by the checks program implemented as described above.

4D Compression

Lorenzo Ibarria
Georgia Institute of Technology

Summary

Contemporary scientific simulations frequently produce large datasets consisting of voxels, representing a 4D volumetric set of values. Such datasets may consume several terabytes of space, and researchers need to inspect them, but usually in practice in only a couple of senses: some high-level views through time, and small sections through time. To facilitate these common modes of interacting with terabyte datasets, we propose a compression method that reduces the size of the dataset, and allows small sections of the dataset to be viewed through time slices.

To address these requirements three different approaches have been tried. All are extensible to 4D, to take advantage of the coherence in time. We tried wavelets, interpolation, and an extrapolating predictor. Of the different options present in the techniques, the predictors work better in the L-infinity metric, while wavelets outperform in L-2 metric. These techniques are complementary, for instance, wavelets do not allow for a small subset of the data to be decompressed without requiring traversing the whole dataset; however, with the extrapolating predictor this can be avoided. The best method we found to compress the corrections of the predictor was a geometric context arithmetic encoder, which is a set of arithmetic encoders indexed by the previous prediction in the dataset.

We plan to extend the scheme to multiple processors, something not easily accomplished because of the propagation of the error in the predictions, and the requirements of the arithmetic encoder. This would allow for a fast compression, and a guided decompression that chooses only the important parts of the dataset, not the whole. This advance is motivated by the new prevalence of multiprocessor machines, and the perspective that there is a lot to gain.

Algebraic Multigrid Methods for Contact Problems in Linear Elasticity

Ana Iontcheva
University of California, Davis

Summary

The reliable simulation of contact problems is of great importance in many applications. In virtually every structural and mechanical system there exists a situation in which one deformable body comes in contact with another. Signorini's problem, describing the contact of a linearly elastic body with a rigid frictionless foundation, is a classical problem and a basis for many generalizations. The construction of fast and reliable solvers is a challenging task due to the intrinsic nonlinearity of the problem – prior to the application of loads to a body the actual contact surface is unknown so a free boundary problem is obtained. Most of the methods derived for Signorini's problem assume that the body that comes in contact has a regular shape so can be approximated by a structured mesh. The problem that we are solving is a Signorini's problem with a body with complex geometry approximated by an unstructured mesh.

The finite element method is used for the discretization of the problem, so the element matrices are explicitly available. We need a method of optimal order for the solution of the discretized problem, namely multigrid methods. The standard geometric mutigrid can be applied only to problems with structured grid, so we are interested in an algebraic multigrid approach. Possessing the element matrices, we can use AMGe (element based algebraic multigrid), but we need to take into account specific difficulties of Signorini's problem – the existence of a contact surface. A special coarsening away from it is appropriate.

Summarizing, we are creating and applying a nonlinear element based algebraic multigrid method with special coarsening away from the contact boundary. The code for the Projected Block Gauss Seidel Relaxation for Signorini's problem and the code for the coarse grid solution procedure – based on Dostal's algorithm – has been developed.

An extension to the visualization tool GlVis for the visualization of the grids created by the coarsening procedure in AMGe has been accomplished in both 2D and 3D. The next step is the incorporation of these codes in the full code for the nonlinear AMGe for Signorini's problem. Our future plans include solving the two-body contact problem.

Accelerating Volume Rendering by Cache Optimization for Graphics Hardware

Ming Jiang
Ohio State University

Summary

The motivation for this work stems from the need to perform volume rendering of unstructured datasets at interactive frame rates. This level of performance is essential for exploring and visualizing large-scale scientific datasets. In order to improve the performance of the volume renderer, we propose a scheme that allows for more efficient usage of the graphics hardware, In particular, we introduce a preprocessing stage to the volume renderer in which the unstructured dataset is first sorted into Hilbert order and then compressed by eliminating duplicate vertices.

At the current stage of development, an unstructured dataset is first sampled, slice-wise, along the X, Y, and Z directions. For each slice, the sampled data points are triangulated using a Marching Cubes algorithm. Depending on the viewing direction, the actual rendering is accomplished by compositing one of the three sets of slices. It is at this point that we introduce the preprocessing stage to improve the rendering speed.

Our goal for this project is to improve the overall rendering speed of the volume renderer by compressing the unstructured dataset and optimizing cache performance for graphics hardware. Our compression scheme is simple yet effective. For each slice, it eliminates all the duplicate vertices generated from the Marching Cubes algorithm. There are two types of duplicate vertices: numerical and theoretical. For most cases, the coordinates of duplicate vertices share the exact same numerical values. However, due to finite precision, the Marching Cubes algorithm can produce duplicate vertices whose coordinates may not match numerically. In order to efficiently eliminate the duplicated vertices without 0(N••2) comparisons, we utilize a 2D hashing scheme in which only vertices within the same hash bin are compared. To ensure that theoretical duplicates are eliminated, we also compare vertices in the immediate neighboring bins as well. The number of hash bins is determined adaptively for each slice, based on the density of the vertices (i.e. number of vertices divided by their spatial extent). Using this method, we were able to achieve 80-85% data reduction.

Our cache optimization scheme involves sorting the triangles on each slice based on the Hilbert order. A Hilbert curve is a space-filling curve that can be drawn from one point on a plane to another without any intersections. Essentially, this curve fills up an entire region before proceeding to the next. In a similar manner, we sort the triangles on each slice so that all the triangles within a region precede the ones in the following region. Using the Hilbert order, we achieve near optimal spatial coherence for the triangles on each slice, which results in near optimal cache utilization for the graphics hardware. In this case, we conducted our experiments on NVIDIA GeForce3 graphics cards.

Currently, all the preprocessing steps are implemented with respect to each slice, which is necessary for the compositing step. However, since the number of vertices per slices can vary drastically, the preprocessing and rendering times can vary drastically as well. To address this issue, we are considering dividing each slice into blocks of fixed size and render the preprocessed blocks instead.

Parallel Algebraic Multigrid Method for Finite Element Problems Based on Domain Decomposition

Tzanio Kolev
Texas A&M University

Summary

The goal of this project was to develop an efficient parallel solver for finite element problems posed on large unstructured grids. Such problems arise naturally in simulations, where only a very fine discretization of the domain is can resolve the physics of interest. In order to achieve performance comparable with multilevel methods for the geometrically refined case, one can use an algebraic method based on sequence of coarsened meshes. Our objective was to develop a parallelization of one such algorithm, namely, the agglomeration-based algebraic multigrid for finite element problems (AMGe).

Our method starts with a partitioning of the original domain into subdomains with a generally unstructured finite element mesh on each subdomain. The agglomeration-based AMGe is then applied independently in each subdomain. It needs access to the local stiffness matrices which are reconstructed after coarsening by the variational principle. Note that even if one starts with a conforming fine grid, independent coarsening generally leads to non-matching grids on the coarser levels. We use an element-based dual basis mortar finite element method to set up global problems on each level. Since AMGe produces abstract elements and faces defined as lists of nodes, the mortar multiplier spaces are also constructed in a purely algebraic way. This construction requires inversion of the local mass matrices on each interface boundary shared between two subdomains. This is possible because of the way AMGe agglomerates the faces. This completes the (non-nested) spaces.

The algorithm was implemented in a general, object-oriented MPI code that uses parts of the HYPRE preconditioning library. Specifically, the program constructs the stiffness matrix A and the mortar interpolation matrix P on all levels and stores them in the ParCSR parallel format. The global matrix for the mortar method is P^T A P and is computed using a “RAP” procedure from HYPRE. We need explicitly the entries of this matrix, since we use parallel Gauss-Seidel with processors coloring as a smoother on each level of the multigrid. Our experience shows that this is a good choice, because its implementation is independent of the spectrum of the matrix. A software framework was developed, where the geometric information provided by a mesh generator is converted to an algebraic one by a problem generator. This is in turn read by the solver, which is independent of the coordinates, the dimension, and the type of finite element basis functions used. Currently, we have problem generators for model two-dimensional and general three-dimensional problems. They run in parallel, refining independently in each subdomain and allowing for general geometry including non-matching fine grids in three dimensions.

A number of tests were performed in order to investigate the properties of the method. Results were obtained on ASCI Blue Pacific running jobs with more than 500 processors.

Tzanio Kolev
Texas A&M University

Summary (continued)

The general observation is that the method is reasonably scalable when the number of processors is increased, while the size of the problem in each processor is kept constant. The setup cost (as with many algebraic methods) is high, but remains bounded as we use more processors. The solution time also scales well if we ignore the time for the exact solve on the coarsest level of the multigrid. The latter starts to dominate for large number of processors and dealing with it through some type of processor agglomeration is a topic of future research.

Another project I was involved in was the numerical testing of negative-norm based least-squares algorithms for div-curl systems. Two-dimensional computations, demonstrating a uniform convergence rate for a method involving finite element space enhanced with “face-bubble” functions were performed. This study is motivated by our interest in robust solvers for the Maxwell equations and is still in its initial stage.

A Discrete Differential Forms Framework for Wave Equations

Joseph M. Koning
UC Davis

Summary

The discrete differential forms framework for wave equations defines a mimetic finite element method for discretizing three-dimensional scalar and vector wave equations.

Scalar and vector basis functions are used to define the discrete differential forms. The method is valid on unstructured grids and conserves all relevant physical quantities such as energy, charge, momentum, and mass. For the lowest order basis functions, the method is second-order accurate in space and time.

Full-wave, parallel simulations in the fields of electrodynamics, linear elasticity, linear acoustics and linear magnetohydrodynamics have been completed. Currently, the research is focused on simulating optical structures such as optical fibers and photonic band gap devices. One of the major accomplishments of this framework is a mimetic discretization of the electrodynamic Helmholtz equation that shifts all zero eigenvalues to calculate the extremal eigenvalues.

In the future the method will be enhanced with higher order basis functions (in progress), h/p-adaptivity and complex materials.

Generating Library-Specific Optimizing Preprocessors using ROSE

Markus Kowarschik
University of Erlangen-Nuremberg

Summary

Large application codes in scientific computing are usually based on the use of libraries, which provide a variety of data structures and functions. In the case of object-oriented applications, these libraries implement high-level abstractions such as array classes, grid classes, particle classes, etc. Unfortunately, the use of high-level abstractions, which are defined in underlying libraries, cannot be optimized by any standard compiler since the semantics of these abstractions are user-defined and therefore unknown to the compiler. This lack of knowledge is a major reason for the poor efficiency of the majority of object-oriented scientific codes: the Mflop/s rates which can be measured at runtime are just tiny fractions of the theoretically available peak performances that the vendors claim for their machines.

ROSE is a software infrastructure for generating library-specific preprocessors which perform source-to-source transformations; e.g., eliminating the need for creating temporary objects, fusing and blocking loops, and introducing several other kinds of cache-based transformations into numerically intensive application codes. Internally, a preprocessor (which has been built using ROSE) parses the original code (currently C++ code) and assembles the corresponding abstract syntax tree (AST). Automatically generated tree traversal routines based on inherited and synthesized attributes are then employed in order to both recognize library-specific high-level abstractions and introduce transformations by replacing old AST fragments by new ones. These new AST fragments are generated by building context-specific code strings and then passing them to the compiler front-end. After the AST has been modified according to the transformations specified by the library writer(s), it is unparsed. This final step yields the optimized C++ source code.

The work during this summer has focused on two issues. The first task was the handling of preprocessor directives such as comments, “include” directives, etc., and their introduction into the generated C++ source code. The problem with these directives is based on the fact that they are not part of the C++ language itself but processed by the standard C++ preprocessor before the actual C++ compiler front-end is called. The second task was to implement the generation of C++ source code which performs cache-optimized array assignment statement transformations for the A++/P++ preprocessor. This preprocessor is used to enhance the performance of applications based on the A++/P++ array library that was also developed at CASC.

Extending the KULL Automated Testing System

Dedaimia Kozovsky
University of Wisconson-Madison

Summary

The Automated Testing System (ATS) is a tool for testing KULL executables. The ATS runs KULL job scripts, then compares the results to previous runs or analytical data to determine if an executable has passed or failed. However, determining correctness is a challenge for two reasons: (1) job scripts do not have a standard type of output or output format – users can have scripts handle output in almost any way they want and (2) there is no single standard for determining when output is "correct". New output is generally compared to some kind of reference data, but this can be done with many different comparison routines – including having a human manually review results. A main focus of development of the ATS this summer was designing a system to overcome these two difficulties. I also added a few other features including a GUI and a new method for storing results and references in the ATS database.

Rather than try to restrict "test" job scripts to certain types of output and predefined comparison routines, the ATS was designed to be extremely flexible by using several libraries:

1) The AtsOutputTypes library defines all types of output that the ATS will accept from a job script. All built-in Python types are in the library, and users can easily add new types.
2) The AtsReturns library defines functions to return output from a test script to the ATS. Output is serialized using Python's pickle module, dumped to standard out, then unpickled by the ATS. Since both the ATS and the test script can access the AtsOutputTypes library, any object type defined in the library can be returned successfully.
3) A third library, AtsComparisons, contains comparison routines that can be accessed either within the job script or by using the "review" command in the ATS. New comparisons can be added to the library at any time. The ATS also allows users to interactively choose comparison routines and set arguments to the comparison.

In addition to the flexibility of these libraries, users can easily change what reference data job results are compared to by adding new references either from a run result or from a file.

Put together, all of these features make the ATS is extremely flexible and extendible. It can handle literally any type of output, as long as the type has been registered in the AtsOutputTypes library, and perform any comparison, as long as the comparison has been added to the AtsComparisons library. Although this amount of flexibility might be overkill in many testing systems, it is essential for thorough testing of a package as complex as KULL.

Future plans for the ATS include: (1) adding more routines to the comparisons library, which is somewhat sparse at the moment, (2) changing the run manager to allow jobs to be run on the batch system as well as interactively, and (3) adding an interface to Perforce to obtain version information on test scripts and executables.

Time-parameterized Contour Trees

Ajith Mascarenhas

University of North Carolina at Chapel Hill

Summary

A common and useful technique for visualizing simulation data is to compute iso-surfaces. Contours of elevations found in topographic maps, where points on the same height on a terrain are shown as curves, are a familiar example of iso-surfaces. We are interested in computing iso-surfaces for volumetric data that is time varying. Such datasets are produced by simulations of physical phenomenon like fluid-flow, mixing of liquids like oil and water, and so forth.

Some questions that come up during visualization of iso-surfaces are how many components are present in the iso-surface? What components merge, split, appear or disappear if one changes the iso-value by a small amount? A contour tree is a useful structure that can be used to answer these questions. We are interested in computing a variant of the contour tree structure that can be applied to time-varying volume data.

Our technique is based on a topological approach to the problem of constructing time-parameterized contour tree. Given the fact that the nodes of a contour tree can be mapped to the critical points - maxima, minima, saddles - of the data, we aim to compute the trace of critical points as a function of time. We call this trace the “gamma” curve. Given the contour tree for time t=0, and the “gamma” curve we aim to compute the contour tree for any subsequent time t=t'.

During the summer we have implemented the construction of “gamma” for a data sampled on a uniform grid. We use a canonical connectivity scheme wherein each grid cell is split into simplices about the main diagonal. We have computed gamma for a few small datasets of size 64x64x64x20 (i.e., a cube of size 64 on a side through 20 timesteps) and plan to work of larger data sets.

In continuation to the work done this summer we hope to develop the theory and algorithms for constructing time-parameterized contour trees. This work will require significant effort during the coming year and next summer.

Detecting and Exploiting Spatial Regularity in Memory References

Tushar Mohan
University of Utah

Summary

Rising processor speed, unaccompanied by corresponding reductions in memory access latency, has caused the performance of codes to be limited by memory accesses. Strided memory accesses, or streams, can be a significant source of memory stalls in loops in practical applications, if undetected. If known to exist in an application, they can be targeted by a host of optimizations, such as stream prefetching, relocation, remapping and vector loads.

Existing stream detection mechanisms either require special hardware, which may not gather enough stream statistics for subsequent analysis, or are confined to limited compile-time detection of array accesses in loops. Formally, little treatment has been accorded to the subject; the concept of locality fails to capture the existence of streams in a program's memory accesses.

In this project we define spatial regularity as a means to depict the presence of strided memory accesses. We develop measures to quantify spatial regularity, and design and implement an on-line, parallel algorithm to detect streams, and hence regularity, in running applications. We identify critical program sections for regularity measurements by using PAPI – a performance measurement API – to access hardware performance counters portably. Dyninst's dynamic binary translation infrastructure is leveraged to perform selective and transitory instrumentation in the application. This allows the user to limit the stream detection overhead at the cost of measurement accuracy. We use examples from real codes and popular benchmarks to illustrate how stream information can be used to effect profile-driven optimizations.

MR-File: A New Multi-Resolution Spatial Index for High-Dimensional Scientific data

Mohamed F. Mokbel
Purdue University

Summary

Existing multi-resolution models for large-scale simulation data can support a wide range of spatial range queries. However, queries about non-spatial variables are not necessarily efficient. Non-spatial variables are stored in the multi-resolution model in a statistical form, where the minimum, maximum, mean, and standard deviation for each variable are stored. The main objective is to build an indexing scheme on top of the multi-resolution model described in to efficiently answer non-spatial queries while keeping the efficiency of the spatial queries.

Our approach is to map the problem into a spatial access method (SAM). The non-spatial v variables in the multi-resolution model are mapped into non-zero sized hyper-rectangles in the v-dimensional space. The state of the art software for spatial indexing, GiST, is used. However, results from GiST are completely unsatisfactory. GiST does not scale with the massive size and the high-dimensionality of scientific data, a phenomenon known as the curse of dimensionality.

Reviewing the literature of spatial indexing methods, it is clear that the curse of dimensionality has not been investigated for spatial access methods. A similar but easier problem, which is indexing zero-sized objects in the high-dimensional space, is well investigated with some solutions for the curse of dimensionality. In this research, we aim to solve the curse of dimensionality for spatial access methods. As a result, a new spatial indexing scheme, termed the Multi-Resolution file, MR-File for short, is proposed. MR-File is scalable in terms of dimensionality and data size.

Parallel Optimizations of Parallel Algorithms

Evan Moran-Bernard
Carnegie Mellon University

Summary

Parallel computing normally calls to mind hundreds of processors working away at some big problem: modeling a galaxy of stars under gravitation, modeling the weather over Kansas, calculating pi to fantastic precision, or analyzing trends in the stock market. Where feasible, parallel computing delivers these results much more rapidly than serial. Why, then, don't all personal computers have several cheap processors instead of one expensive one? Why can't one's palm pilot link to every palm in a room, rather than just one at a time? In general writing parallel programs is more difficult, and takes more time. Our main interest in this project is to optimize parallel code: to make it be "more parallel."

Making programs "more parallel" involves several levels of parallel-algorithm design. Within the file system I developed parallel algorithms to sort and parse enormous files, and to distribute the data to an arbitrary number of processors. To handle per-processor balancing issues I used several randomizing techniques, which appeared to be successful and scalable. Several constraints occurred at the per-system memory level, where it was necessary to keep a minimal amount of data in memory during the sorting process. Specifically, I used an Oct-Tree data structure, well known in graphics research for improving rendering performance, to spatially sort files and to improve locality in the parallel algorithms. At the distributed systems level I used MPI, the Message Passing Interface, to allow inter-process communication. This, in particular, became a conceptual hurdle because of the inherent difficulties with deadlock and test case reproduction.

Additional efficiency in programming parallel systems may reside in more automated time analysis techniques used to analyze time performance. This information would be useful for finding performance bottlenecks in working code. Analysis also informs the programmer of how close to the theoretically best performance his or her program has achieved. This is perhaps not as important to performance as adding more processors, but the potential is definitely interesting.

Parallel AMG Performance Issues

Arne Naegel
University of Heidelberg

Summary

Driven by the need to solve linear systems arising from problems posed on extremely large, unstructured grids, algebraic multigrid (AMG) methods have been recognized as an efficient way to treat sparse systems of the form Ax=b. The advent of high performance computers with large numbers of processors sparked interest in parallel versions of AMG that are capable of employing the available sheer computational power to solve large systems of equations in a robust fashion. Although multigrid methods in general are known for optimal (i.e., O(n)) performance in the solution phase and AMG particularly shows fast convergence rates for a wide variety of problems, in general, results are still sometimes dissatisfying. The reason is that the non-geometric approach — due to the lack of knowledge about an underlying grid structure — requires a setup phase that is often remarkably expensive, and its performance usually degrades further in a parallel environment.

Abstracting the implementation of the BoomerAMG, the parallel AMG solver integrated in the software package hypre (High Performance Preconditioners), we derived a parallel computing model for the setup phase and described the performance of its three major components in terms of the degree of the related matrix graph, the number of unknowns, and of the processors involved. As it involves critical message passing components, one key part was the analysis of one of the coarsening strategies used. Additionally, several numerical tests were run and its results were compared to the developed theory.

The aim of this work was to analyze the behavior of the AMG setup phase in a (massively) parallel environment on a distributed-memory architecture like ASCI Blue Pacific at LLNL, and to gain insight as to those stages of AMG code / algorithm where improvements can be achieved. The close collaboration with Rob Falgout and the Scalable Linear Solvers Group at CASC assures that the results will lead to setup phase improvements in the future.

Regression Testing for Lustre's Distributed Locking Subsystem

James Newsome
University of Michigan

Summary

Lustre is a high performance distributed file system being developed by Cluster File Systems. The project is being assisted by Lawrence Livermore, Sandia, and Los Alamos labs so that it may be used in the next generation of supercomputing clusters.

One of the innovations in the Lustre file system is a distributed lock manager. Rather than having a central server that each node must contact to lock resources, each node in the cluster is capable of managing resources. This has two distinct advantages: First, it eliminates the lock server as a single point of failure. Second, by allowing the node that uses a particular resource the most to manage that resource, we eliminate the need for many lock requests to be sent over the network.

Thorough testing is extremely important in any software project. It is especially important in a system as complex as Lustre's distributed lock manager. My role in the project was to develop a regression test that would verify that the locking subsystem works correctly under a large volume of requests.

One goal of the test was to make it as independent as possible from other parts of Lustre. If the test made heavy use of other parts of Lustre, it would be more difficult to find the source of a failure.

Another goal was to balance testing as much functionality as possible with making it easy to identify problem areas. In a test that tests everything, passing the test assures us that everything is working correctly. However, failing such a test does not provide much information in itself, since it could be any part of the locking subsystem.

In order to isolate the test from other parts of Lustre, it was put inside the Linux kernel. This is currently the only way to directly access the locking subsystem. Another option would have been to have a user-space test work with filesystem objects, putting various resources under contention. However, this would defeat the goal of isolating the test from the rest of Lustre. The test would be dependent on the whole system rather than just the locking subsystem.

The test exercises the following functions of the locking subsystem:
– Enqueueing an extent lock
– Matching an extent lock
– Using completion callbacks to keep track of locks
– Using blocking callbacks to cancel locks

The test first starts some number of threads. Each thread then runs in a loop. In each iteration of the loop, each thread randomly either tries to match/enqueue a lock on a

James Newsome
University of Michigan

Summary (continued)

random extent of a random resource, or decrement one of the lock references it already holds. The test runs indefinitely until a user stops it. Short runs of the tests, on the order of several hundred enqueues and matches, pass. Unfortunately, I have observed some strange behavior when the test is run longer. It is currently unclear whether this is caused by a bug in the lock manager, or a bug in the test itself.

For the future, the first thing that needs to be done is to find what is causing the test to fail on long runs. Once this is done, there are several enhancements that could be added to the test.

First, the test should also test lock conversions. The code for this is partially written, but I left it disabled for now in favor of getting the simpler version of the test to work. Once this is added, the test will be able to verify the functionality of all the basic parts of the distributed lock manager.

The next step after that would be to make the test more distributed. Currently it is designed to run only on a single node. A more realistic and thorough test would be to have several nodes all trying to work with the same set of resources.

After these features are added, I believe the test would be complete. The only other thing to add to the test would be boundary conditions, such as cancelling a lock that hasn't been granted yet, or having a node fail during the test.

Multigrid for Linear Hyperbolic PDEs

Luke Olson
University Of Colorado at Boulder

Summary

Multigrid is firmly established as the method of choice for systems of elliptic and parabolic partial differential equations and the Laplacian-type terms that dominate the ill-conditioning of more general systems of PDEs. Many practical codes have been deployed in ASCI and other applications based on multigrid methods, leading to increased desires to extend multigrid to additional regimes. The First-Order System Least-Squares formulation extends multigrid theory to many such systems. However, the purely hyperbolic case has received relatively little attention (or successful attention) from the multigrid community.

In this project we consider First-Order System Least-Squares formulations for purely convective partial differential equations and analyze the multigrid perfor-mance for solving the linear system arising from a finite element discretization. Prior research exposed poor multigrid convergence results under certain boundary conditions with Algebraic Multigrid (AMG). Local Mode Analysis (LMA) was applied to the resulting finite element stencil and we explored a geometric multilevel framework to further reveal the unexplained inefficiencies in the solver. The LMA predictions of relaxation and two-grid performance were then compared to the actual implementation of the Geometric Multigrid algorithm.

Through our analysis and numerical tests, we found both poor smoothing properties and unsatisfactory coarse-grid approximations to cause the degraded performance. Based on these results, we proposed adjustments on improving the smoothing and coarse-grid components of the multigrid algorithm. Still, these new algorithms need further investigation. This research will be continued at the University of Colorado, where we intend to address issues such as scalability and robustness of the method.

A Hierarchical Shrink-Wrapping Approach to Surface Reparameterization

Serban Porembescu
UC Davis

Summary

Our goal is to develop a robust, efficient, and easy to implement surface reparameterization technique. Parameterized surfaces are amenable, among other things, to Continuous Level of Detail Control, Multiresolution Editing, Texture Mapping, and most importantly, Compression.

Our technique begins with an arbitrarily triangulated surface and constructs a hierarchy of fine resolution to coarse resolution surfaces. As the hierarchy is constructed, mappings between adjacent resolutions are maintained.

We reparameterize the surface by applying, in combination, a set of atomic operations. These operators are Subdivide, Smooth, and Snap. The Subdivide operator refines the base mesh by performing bilinear subdivision. This gives our surface a quadrilateral grid-like structure. The Smooth operator minimizes the mesh distortion by trying to make all quadrilaterals equally sized. Using the Snap operator we can project the base mesh onto any level of the hierarchy.

Given these atomic operators we construct a set of Super operators: Smooth-Snap and Subdivide-Snap. The Subdivide-Snap introduces new vertices in the base mesh and projects these vertices onto the current hierarchy level. The Smooth-Snap reduces distortion by pulling vertices off of the current level of the hierarchy and then projecting the modified vertices back onto the current hierarchy level. The end result is a reparameterization of the original surface.

To make this algorithm truly useful for large-scale data visualization we need to make the algorithm work out-of-core. Extremely large data sets are being generated by improved resolution scanners, sensors, and computer simulations. Often this data is too large to fit in the main memory of a high-end workstation and in some cases even in the main memory of the supercomputer that generates the data. The current algorithm has been designed to generalize to out-of-core implementation.

High Order Discrete Differential Forms for Computational Electromagnetics

Robert N. Rieben
UC Davis

Summary

We are concerned with the finite element solution of Maxwell's equations on unstructured hexahedral grids. We focus our attention on the high-order discretization of this problem, specifically the recently proposed differential forms based approach for constructing curl-conforming and divergence-conforming vector bases. In the computational electromagnetics (CEM) community, it is well known that in order to accurately perform electrically large simulations (i.e., simulations whose characteristic dimension is many wavelengths long), high-order methods must be used. Low-order finite difference and finite element methods are ineffective for such problems due to numerical dispersion. Recently, several advancements have been made in the field of (what are presently called) Vector Finite Element Methods. While it is known that in general higher-order methods have less dispersion and should be superior to low-order methods, this has not yet been demonstrated for Vector Finite Element solutions of Maxwell's equations. In addition, there are few high-order implementations of this method; and of these, none present a general technique for the construction of hierarchical bases, which are necessary for any sort of p-refinement strategy. In conjunction with my thesis project, my research this summer has been the development and implementation of a method for solving Maxwell's equations using Discrete Differential Forms (DDF's) of arbitrary order on unstructured hexahedral grids.

In the language of differential forms, we can unify several existing finite element methods using a clear and concise notation. Traditional finite elements (scalar nodal) as well as vector finite elements (edge and face elements) and discontinuous scalar elements are combined into one set of discrete differential forms. We begin by constructing DDF's of arbitrary order on a reference (or unit) hexahedron. We then define a set of linear functionals (referred to as Degrees of Freedom) that map the DDF basis functions to real numbers. These are necessary for the construction of element mass and stiffness matrices. A major accomplishment of this summer was the development of a general method for constructing hierarchical vector basis functions using an appropriate set of Degrees of Freedom. A hierarchical basis is necessary for any p-refinement method. DDFs on the reference element are then transformed via conformal mappings to global mesh elements of arbitrary location, orientation and distortion.

Over the summer, the mathematical framework of this method was developed and implemented in the FEMSTER framework. FEMSTER is a modular finite element class library for solving three-dimensional problems arising in electromagnetism. The software consists of a set of abstract interfaces and concrete classes, providing a framework in which the user is able to add new schemes by reusing the existing classes or by incorporating new user-defined data types.

Future work will include the integration of the FEMSTER library into a general parallel matrix assembly program. Once this is complete, work will begin on a time stepping algorithm in conjunction with the HYPRE library for time domain simulation of various electromagnetic transmission devices such as optical fibers and Photonic Band Gap devices.

A Web Services Framework for Bioinformatics

Daniel Rocco
Georgia Institute of Technology

Summary

The transition of the World Wide Web from a paradigm of static web pages to one of dynamic web services provides new and exciting opportunities for bioinformatics with respect to data dissemination, transformation and integration. However, the rapid growth of bioinformatics services coupled with non-standardized interfaces diminish the potential that these web services offer.

For instance, the BLAST family of bioinformatics data sources allow biologists to compare DNA and protein sequences with an existing body of knowledge to find similar sequences in other organisms. BLAST provides data management and query services to genomic data sources and allows research groups to set up local repositories or specialized sequence databases. One catalog lists over 500 such data sources, some subset of which will contain data relevant to an arbitrary query.

Unfortunately, there is no common interface or data exchange mechanism for these sites. When performing a search, the scientist chooses a set of known sites, enters a query into each site, and integrates the result by hand.

The problems with this approach are numerous: the scientist may not have the most current or most relevant site for the search at hand, the search must be entered multiple times, and the results of the search must be merged together by hand to obtain an integrated set of results.

Our challenge is to provide a common interface to the vast and dynamic assortment of BLAST data sources. This problem can be divided into three sub-problems. First, how will new sources of relevant information be integrated into the interface? Next, how can the interface determine which discovered sources are relevant to a particular query? Finally, how can the interface mediate between sources and produce an intelligently integrated result set?

We propose the notion of a service class description as a solution to the problem of discovery, classification, and integration of bioinformatics sources. A service class description describes the relevant portions of the service from the point of view of the intended application. The description includes the various data types used by the service, example queries and output, and a graph representation of how service class members are expected to operate. For example, a simplified view of the DNA sequence BLAST service class includes a DNA sequence input type, a BLAST result output type, and descriptions of the intermediate pages. The control flow graph might show the input page being connected to a result page, possibly through a delay page.

The service class provides an abstract view of bioinformatics sources that allows developers to reason about the class rather than being concerned with the intricate details of each member. Using the service class description, we are creating an integrated BLAST service that will operate via a common application to an automatically discovered set of BLAST sources, each wrapped to provide a transparent interface between the application and the data source.

The Generic Simulator

Sadik Gokhan Caglar
University of San Francisco

Summary

The GenSim project is a generic simulator, that is mainly used to simulate a cluster of machines that do distributed computing using message passing libraries. GenSim is developed under C++ using object oriented techniques, for the purpose of reusability, modularity, and maintainability. The reusability principle also goes hand-in-hand with the generic nature of the project.

GenSim is a multi-layered simulator. At the very bottom level, there are seven different classes. These base classes provide the general framework on which the simulator runs. From a different perspective, these base classes can be seen as the engine of the simulator. The different layers on top of this layer inherit properly to model specific applications.

The naming scheme of these base classes might suggest that the framework is designed to run only cluster models. In fact, other environments, such as a distributed database, or a specific architecture can be modeled with the base classes, by implementing appropriate interface layers.

The project currently has two layers implemented. The first layer is used to model a programming language behavior. It provides tools like conditionals, deterministic and probabilistic loops, system messaging, creation and termination of jobs, and computation.

A system heap for the variables is not yet implemented, so the conditionals are only probabilistic for the time being. We are able to have deterministic loops, due to the internal structure of the loops. The statements (events) that are in the loop have no idea about the current state of the loop. For instance, if a given loop body executes three times, a computation event has no idea to which execution the current one belongs; only the loop conditional event is aware of it.

The second layer is a subset of MPI, the message passing library. The layer provides means to model blocking and non-blocking communication, waiting for non-blocking communication, and collective communication. Blocking communication is modeled using basic send and receive events. The events use a tag, and for every send event there is a matching receive event in the peer, and vice versa. Non-blocking communication works the same way; however a wait must be called in order to make sure that the communication has been completed. This forces a rule on our system. The loop bodies have to be sound in terms of the non-blocking events they have. In other words, they have to call the matching waits before restarting the loop. If the loop is restarted without knowing that the non-blocking communication has been terminated successfully, the action may cause the event to be discarded. Collective communication

Sadik Gokhan Caglar
University of San Francisco

Summary (continued)

in its nature is blocking, so it doesn't suffer from this possibility. Since all of the collective communication routines have the same requirements for execution (that is, all the processes in the communicator call the same collective, with the same parameters), they can be implemented in the same way. The differentiating property is the type of the collective communication, which is handled by the tags, and the amount of time it is likely to take to run.

Since in real runs the order of collective operations is not always fixed, a random number generator package, randlib.c from University of Texas is integrated into the system. The computational, collective communication and point-to-point communication events take advantage of different distribution functions of this package.

There are two front ends that the users can use to model algorithms; they can use C++ directly and link with the existing classes, or they can use the GenSim runtime environment. The latter is encouraged. The Jacobi iteration algorithm is implemented as an example how the suite can be used to model algorithms for the runtime environment.

The two layers are examples of the capability of the simulator. New classes can be inherited from the existing classes to achieve more functionality. The runtime environment has to be updated with the possible newly added clusters, events, constraints, messages, jobs or processors.

Transformations for the Cache Optimization of Applications using Object-Oriented Abstractions

Sunjeev Sikand
UC San Diego

Summary

A common problem with object-oriented C++ scientific computing is that the high level semantics of abstractions introduced (e.g., parallel array objects) are ignored by the C++ compiler.

ROSE is a programmable source-to-source transformation tool for the optimization of applications using high-level abstractions commonly found in C++ object-oriented frameworks. In our work we target the A++/P++ array class library and implement three cache-efficient optimizations. These transformations exploit both the spatial and temporal locality of memory references exhibited by stencil statements. Experiments have shown a performance improvement of 1.5 to 2.2 time beyond that of previous transformations of array statements into lower level C code associated with the stencil operations (which provided a 2 to 4 times improvement). While the C code transformations are relatively complex, the cache-based transformations are beyond the complexity that is reasonable to expect programmers to write by hand. Thus we present the case that the use of the semantics of high-level abstractions can be expected to lead to faster code than what can likely be produced by hand.

Our results are encouraging and leave the possibility open for further refinement of these transformations to achieve greater performance gains.

RVR, A Genome Comparison System

Jonathan Strasser
UC Davis

Summary

Within the human genome project, it is important to see the similarities between genomes of different species. The most common approach to do this is to perform a BLAST search of one genome against the other. There are two main problems with this approach, both of which come from the fact that the entire genomes are very large. The first problem is that the searches will take weeks or even months. The second problem is that the computer may run out of memory and crash. Our objective is to develop a system that will not have these problems, but at the same time give us the same results.

Sam Rash and Paramvir Dehal came up with the main idea behind our approach. This is to create a map of regions on one genome to regions on the other genome. In our case we wanted to map regions on Fugu Rubripes to regions on the human genome. To do this we decided to first map the IPI protein set to both Fugu rubries and human. Since this a much smaller data set, the comparisons can be completed in a much smaller time period. We were also able to make use of a program called BLAT, which performs BLAST-like searches in less time. In our case we were able to complete these maps over the course of a few days. Once both maps were complete we used the transitive property to finally map Fugu Rubripes to human. Now we can perform a BLAST search using much smaller regions rather than entire genomes.

I implemented this method through a series of scripts, written in perl, that I call the RVR (region vs. region) system. It contains eight main scripts along with four subscripts that take the user through every step of the process. From the map creation using IPI to the running and parsing of region BLAST searches. The RVR system takes advantage of Sam Rash's job distribution program and is able to split the process into thousands of jobs and send them to the JGI servers.

After using this system several times, I have determined that this process can be completed in around a week using the RVR system. The results of the region BLAST were a mixed blessing. On one hand they had much less noise (bad hits that mean nothing and take up space), but on the other hand there were some hits that the RVR system had clearly missed. This can be expected, however, since regions that did not map to an IPI would have not been included in the final BLAST. Since a large amount of time and computing power is saved, the slightly sub par output is acceptable.

Now that the system is complete and tested, it will be used with future versions of Fugu Rubripes and human. It may also be used to compare other genomes as well. The RVR system will also continue to be debugged and tweaked if needed.

Multi-Processor Isosurface Calculation

Mark Stuppy
University of Missouri-Rolla

Summary

The calculation of an isosurface on a single computer is a time-consuming task. The time required to complete this task can be greatly reduced by splitting the work up amongst multiple processors. The objective of this summer's project was to implement a multi-processor isosurface calculation program and test its efficiency and scalability.

The program itself was written using many different languages that worked together to create the final output. Interpreted Python, with the pyMPI and NumPy modules, was used to set up the environments on each processor and to pass messages between processors, a combination of compiled Python and C++ code was used to do the actual isosurface calculation, and the Yorick visualization tool was used to view the isosurface.

To make the original isosurface calculation process work on multiple processors in parallel, each processor was given a small portion of the full problem space to work on. Message passing, using the pyMPI module, was used to send necessary boundary cell information to adjacent processors. The results from the calculation were then dumped to multiple files (one for each processor), which could be read into and viewed by Yorick at a later time. The resulting program ran in a time which is inversely proportional to the number of processors used in the calculation.

Future improvements include finding more efficient data structures in which to keep intermediate information, finding a less-restrictive scheme for passing multiple messages at the same time, and finding a more structured file format in which to create dump files.

Embedded Boundary Technique in Structured Adaptive Mesh Refinement CFD

Ryuta Suzuki
University of Minnesota

Summary

The embedded boundary technique is a means of extending structured mesh discretizations, including adaptively refined structured meshes, to completely general geometries, such as those encountered in computational fluid dynamics (CFD). Motivated by the current development of micro- and nano-technology in aerospace applications, we implement a Navier-Stokes solver for incompressible flow simulation. Efficient computations of this type of flow are increasingly important aids in the design of micro- and nano-scale vehicles.

We implemented the incompressible Navier-Stokes solver in an adaptive Cartesian mesh setting. In order to handle the incompressibility constraint we used the projection method. The projection method is based on the Helmholtz-Hodge decomposition theorem where the arbitrary velocity field with appropriate support and no-slip boundary condition is decomposed into an incompressible velocity field and the gradient of some scalar function. The scalar function is computed from the Poisson equation with Neumann boundary condition and it well approximates the pressure field we seek. The Poisson equation solver is the most intensive part in incompressible flow simulation. Brian Gunney in the Center for Applied Scientific Computing (CASC) implemented the Fast Adaptive Composite Grid (FAC) Method for the Poisson equation in Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI) framework. FAC achieves fast convergence of the iteration using the multigrid philosophy on composite grids.

One of the drawbacks of the Cartesian mesh is its difficulty in handling the boundaries that are not aligned on the meshes. Thus, it is important to develop the embedding boundary technique to handle more complex geometry in structured meshes. We employed the so-called “masking” technique to embed impermeable wall boundary conditions. Based on a no-slip and no-flux boundary condition, we interpolate the velocity and flux to satisfy the above two conditions. Although this is a rough approximation it works well thanks to the multi-level meshes provided by SAMRAI.

Future work will be dynamic adaptation, utilizing the SAMRAI adaptative capabilities. Devising an a posteriori error estimator for goal-oriented adaptation is also an important issue.

An Implementation of ARCHES into a Spatially Adaptive Framework for Pool Fire Simulations

Jeremy Thornock
University of Utah

Summary

The solution of a hyperbolic partial differential equation may be achieved by casting the spatially semi-discretized equation as a high-dimensional ordinary differential equation system through a method of lines strategy. Time integration may then be performed with various standard techniques. However, some methods prove unstable when discontinuities or high gradients in the field are present. This can result in overshoot, oscillations, and general instabilities. One cure for such problems lies in a class of time integrators, namely the Runge-Kutta "Strong Stability Preserving" (SSP) integrators. This type of integrator maintains the stability properties provided by a forward first-order Euler explicit integration yet increases the time accuracy without instabilities.

The main objective was to implement a Runge-Kutta SSP time integrator in a spatially adaptive framework, SAMRAI. This included validation and verification of the integrator within the framework using both adaptivity and non-adaptivity to ensure correct implementation. Upon successful completion of this task, a CFD code (ARCHES) was to be implemented into the adaptive framework using the SSP time integrator with an implicit pressure solve.

In order to simplify the solution procedure, time refinement was not used when implementing the Runge-Kutta SSP integrator. The time step was calculated as the minimum of all time steps over all levels and patches, which met with the Courant-Friedrichs-Levy (CFL) condition. This simplifies the algorithm by negating the need for flux-corrections at coarse-fine interfaces. This approach can be justified quantitatively through experience of those familiar with spatially adaptive algorithms. In general, it has been found that the bulk of the calculation in an adaptive environment is spent on the more refined levels, especially when a higher refinement ratio is used. Therefore, little efficiency is lost in stepping the coarser levels at the small time steps. No study was performed here to validate this conventional wisdom.

The algorithm was tested using Burger's equation with Riemann initial conditions and the linear advection equation using several different initial conditions. The MUSCL scheme was used to discretize the spatial elements of Burger's equation while a monotonicity-preserving scheme (MP5) was used for the linear advection problem. They were both compared with an equal order, non-SSP time integrator. The solution was also verified using the theoretical wave speed to mark the solution for refinement. In other words, the cell tagging routine was in no way connected with the state of the solution. It was found that the integrator performed well in both an adaptive and non-adaptive environment. It was also found that the solution followed the theoretical wave speed exactly, both with and without adaptivity. A write up on the subject of the time integrator was produced and can be referenced for more detailed information.

Jeremy Thornock
University of Utah

Summary (continued)

The next step involved implementing ARCHES into the SAMRAI framework using the same time integration technique, again without time refinement. This is work in progress. The Navier Stokes equations present a special challenge when dealing with a multilevel calculation due to the link between the pressure and mass preservation. It is proposed to use a canned FAC solver that is available in SAMRAI to tackle this issue.

Future plans include finishing the ARCHES/SAMRAI code and testing it with a simple, laminar helium plume case. This should help validate a successful implementation of the scalar transport equation, mixing model, pressure solver, and momentum update using the Runge-Kutta time integrator. Turbulence models are to be added later.

Using a unique formulation which is evolving through the CSAFE project at the University of Utah for handling multiphase flow, it is also proposed that research into so called “cut-cell” methods are investigated using the ARCHES/SAMRAI formulation. Current studies include using such a formulation to model reacting, multiphase mixing tanks.

Distributed Linda Tuplespace

Aaron Wegner
Baylor University

Summary

We tested a Distributed Linda Tuplespace implementation called OpenLinda with a small Monte Carlo simulation. OpenLinda proved to be a weak implementation in that scaling to use multiple machines was very inefficient. We therefore wrote a new Distributed Linda-style Tuplespace implementation with better scaling ability. We began with Postgres Database as the server, but later adopted the the more efficient MySQL database.

The Linda tuplespace technique is a simple way to do distributed computing. Shared data is stored in the tuplespace, and other computers connected to this distributed tuplespace can access it. The three main functions by which Linda is defined are:

set() - insertion of a tuple into the tuple space

get() - destructive retrieval of a tuple from the tuplespace

read() - nondestructive retrieval of a tuple from the tuplespace

My Distributed Linda Tuplespace API also includes getall() and readall(), which allocate and return large arrays of all matching tuples rather than just a single one.

Each function in the API can take any number of arguments up to 256. Types accepted are int, long int, double, char*, and BLOB. The BLOB type is primarily used for storing other types of data than the standard types, such as user created structures. The general format for each function is:

func(TSconn *conn, char *types, var, var, ...)

- where *conn is a pointer to a connection of type TSconn to the Linda Tuplespace.

- *types is a string specifying the types of the following variables, and

- var, var, ... is the list of variables to get or set.

The syntax for the specifying data types in the types string is simple. For each variable, put a '%' followed by the first letter of the desired data type. For example, to insert a tuple with types 'int, int, double, blob', the types string would be

Aaron Wegner
Baylor University

Summary (continued)

"%i%i%d%b". It is also acceptable to use the full data type, delimited with %'s. "%int %int %double %blob" would work fine. For return values, precede the type with lower case 'r'. "%i%ri%d%rb" will return and int and a blob for a tuple that matches the given int and double.

A MySQL database must be installed to use this Linda implementation. The Linda Tuplespace must have privileges to select, insert, update, delete, create, drop, and index.

The main limitation of the efficiency of this software is the database backend. Triggers and stored procedures are not supported in the current version of MySQL (v4.0). Once these are implemented, Linda could be modified to use them to run somewhat faster. Additionally, it should be fairly easy to adapt this source code to take advantage of a different, more efficient database than MySQL. One more possibility would be to write a tuplespace server from scratch that does not have the overhead of a relational database. In the future, I would like to work out some solution to get better performance out of the tuplespace server.

I would also like to add XML support to enable using the tuplespace across multiple platforms simultaneously. Data would be encapsulated into a standardized format and viewable from any machine. For functionality, it would be nice to be able to specify exactly how many tuples to grab from the tuplespace instead of just being able to get one tuple or all tuples. It also might be possible to take advantage of the high efficiency of getall() to improve the speed of get().

This new implementation of a Linda Tuplespace is a success, both in and of itself, and as a step along a development path. Since it handles an arbitrary number of connections and uses BLOB data, it is versatile. It scales well because the main limitation to scaling is the database backend used. MySQL does a good job of handling many connections, and a fairly good job with speed in relation to comparable databases. This Linda Tuplespace software would be most useful on a system that does a large amount of computation and has many connections to the tuplespace.

Building a Prototype System for the Automation of the Acquisition of Earthquake Bulletins

I-Hsuan (Sandy) Wu
UC Davis

Summary

To support ongoing research, the GNEM program maintains a large and growing database of seismic waveform and parametric data. Although we have largely automated the process of loading data into the database, acquisition of the data still requires nearly full-time attention of two people. Since much of the data are available electronically, we intend to automate a large portion of the data acquisition. As a first step, we have decided to automate the acquisition of earthquake bulletins. We have decided that the system must be distributed among several computers. However, we have not settled on an implementation language or on a technology for handling distributed objects. This summer's project was to build a prototype FTP agent that could be used by the system to automatically acquire earthquake bulletins and forward them to a processing queue. This served as an experiment to help us evaluate the implementation difficulty and efficiency of available technologies.

Several programs were written to compare the capabilities and functions available in C++ and Java for performing FTP connections. Then, we did some research into the available technologies to determine the best way to acquire a distributed system. In the end, Java was selected as the programming language to be used in order to take advantage of the Java Remote Method Invocation (RMI), which enabled distributed programming. The FTP package used is NetComponents. The resulting prototype is a stand-alone system that runs an http server to receive RMI requests from clients. The system retrieves the requested data from a remote location with FTP and returns the result to the client. By using Java, the system becomes portable. And the RMI allows data requests to be made over networks and does not require the client to reside on the same machine with the FTP agent.

Based on the work accomplished, we have decided to implement the FTP agents and their manager using Java and RMI. This will involve evolving the prototype to forward error handling to the agent manager and to process additional messages that the final system will require.

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551